VOMS from a to z

Meant to create coherency

Oscar Koeroo

index

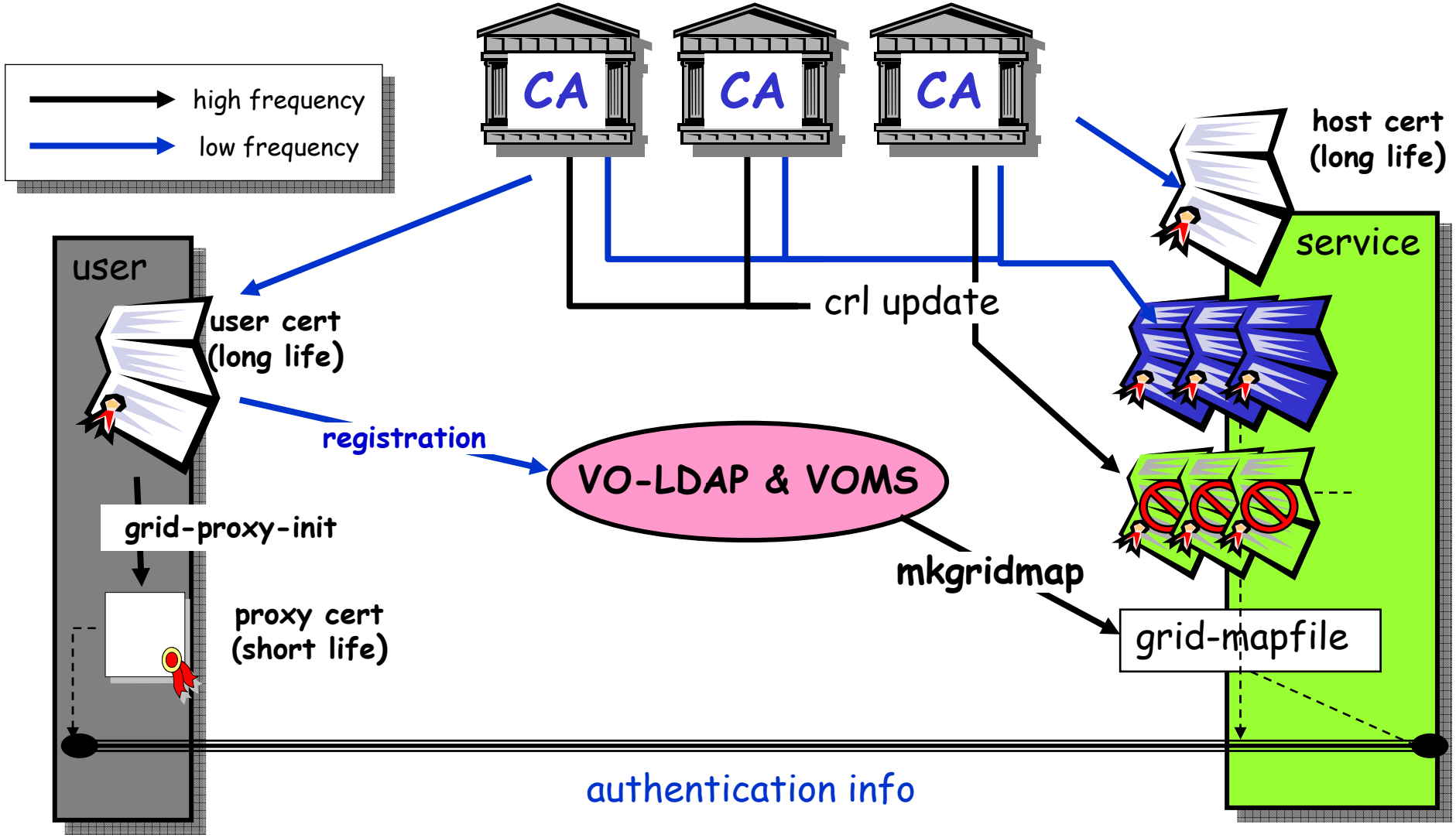
- **The big picture**
- **Registration**
- **In-depth view on VOMS**
- **Sequences**
- **A vulnerability & solution**
- **The distribution problem**
- **The VO Naming Convention**
- **AuthZ & Mapping**
  - The Tools
  - LCMAPS: How we perform a mapping
  - Examples
- **GUMS vs. LCMAPS**

Time conserving rules…

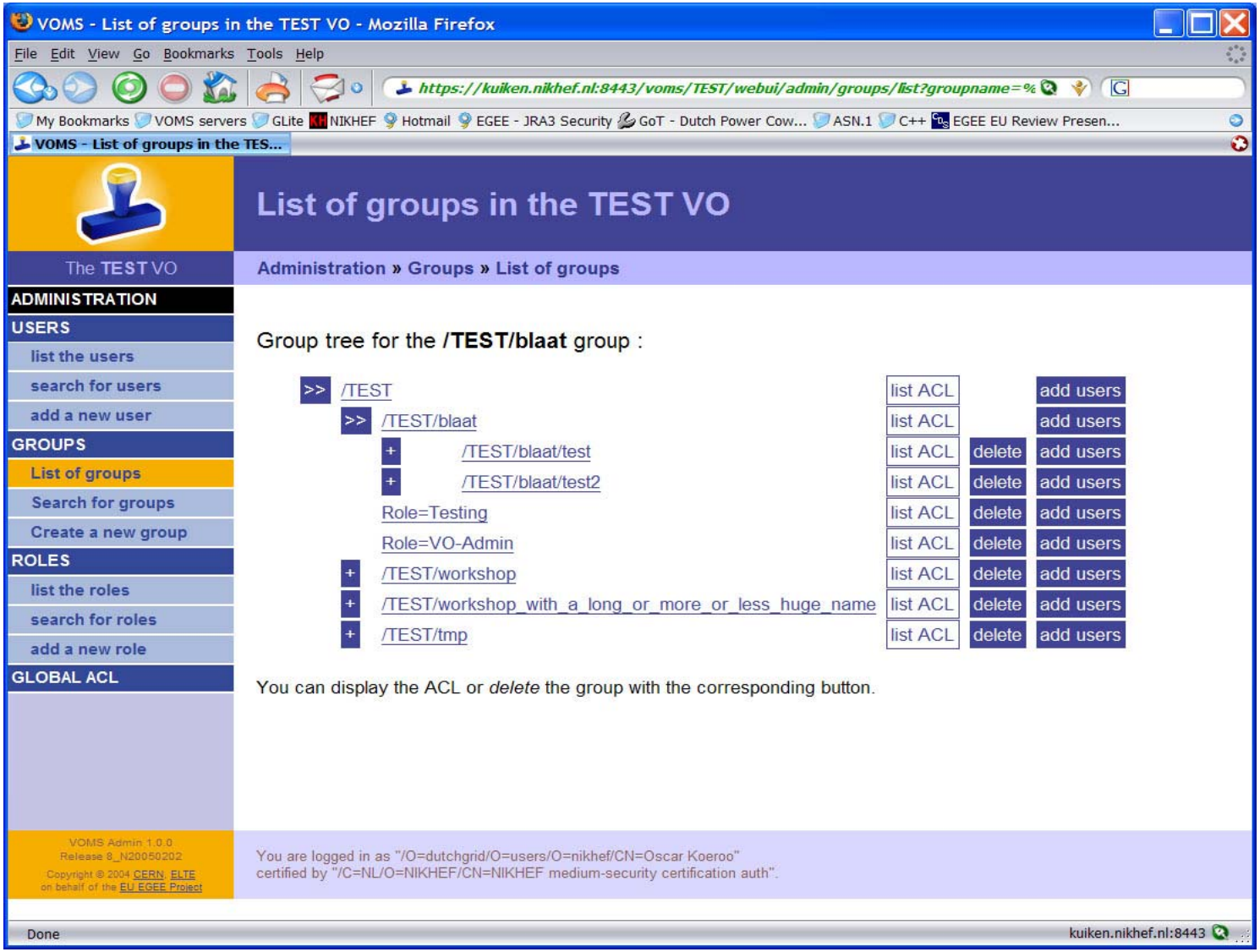
- **Watch for the ’?’ for questions on the end of each section**
- **Interrupt me ASAP for:**
  - (near real-time) corrections
  - if I’m going to fast (or to slow)
  - If you can’t understand crucial info.
- **Discussions planned at the end**
  - … so let’s not waste to much time on discussions during the slide-ware ☺

The big picture

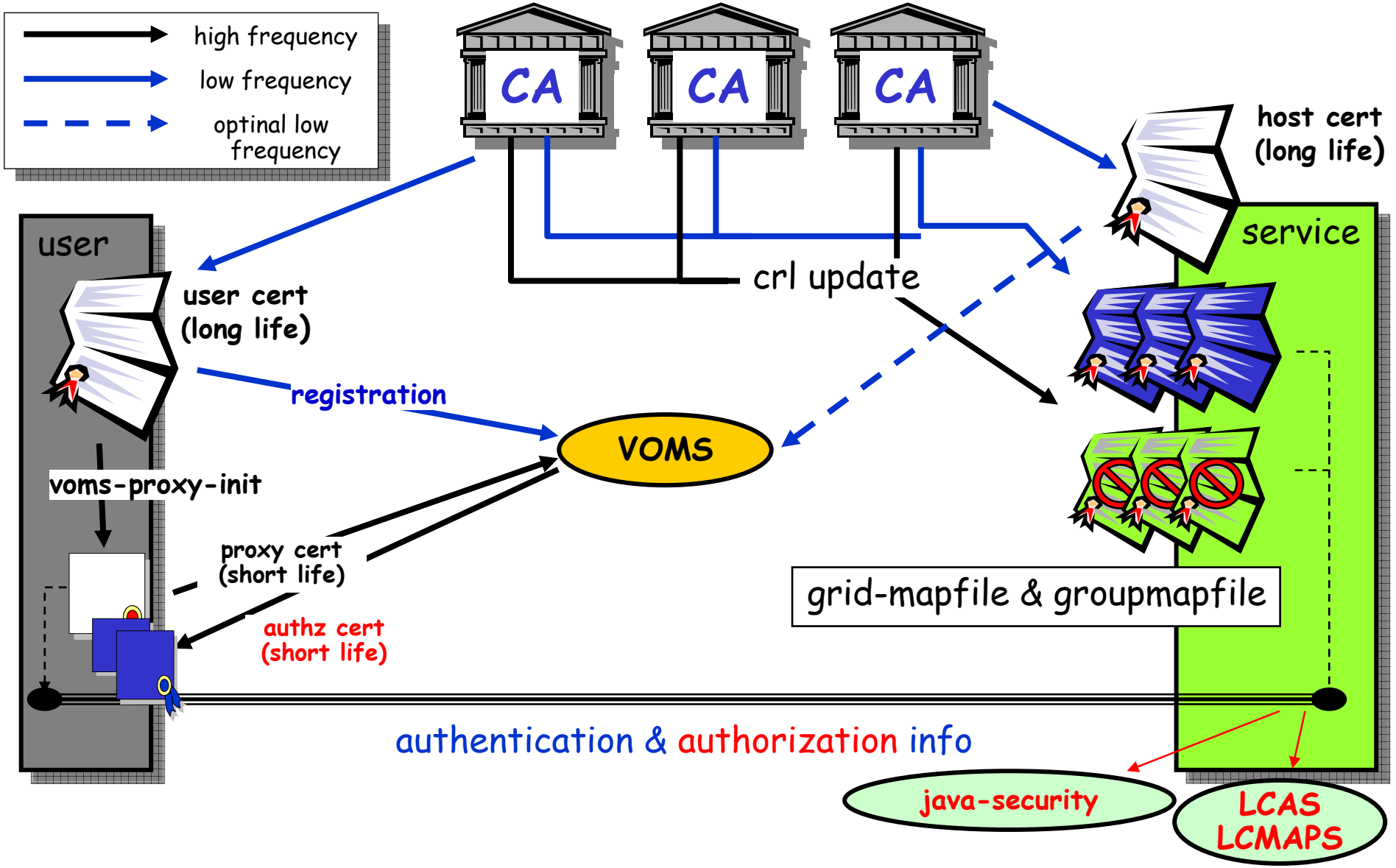
Traditional grid-mapfile



VOMS adds Group and Role structure



VOMS


high frequency
low frequency
optinal low frequency
CA
CA
CA
host cert
(long life)
service
user
crl update
user cert
(long life)
registration
VOMS
voms-proxy-init
proxy cert
(short life)
authz cert
(short life)
grid-mapfile & groupmapfile
authentication & authorization info
java-security
LCAS
LCMAPS

Registration

Registration process

- **Using any form of online registration**
  - VOMS Admin
  - VOMRS
  - By hand…
- **Using TLS or OpenSSL (CLI or online) to get the DN (and CA) of the user**
  - Registering that DN with the VO
  - E-mail exchange
- **VOMRS has the ability to include external services (like databases) in the registration process**

In-depth view on VOMS

In-depth view on VOMS (1)

- **AC as defined by RFC 3281**
  - VOMS OID: 1.3.6.1.4.1.8005.100.100
  - To prevent the stealing of VOMS ACs and other sec. measures:
    - DN of Attribute Holder linked into the ACs
    - Serial Number of User Certificate linked into the ACs
    - ACs have their own Validity period
  - ACs are signed by the private key of the VOMS Server Host certificate
    - Nothing prevents the use of a service certificate or user certificates instead of host certs in this signing process

- **The Authorization tokens are listed as *FQANs* in the AC**
  - FQAN: Fully Qualified Attribute Name
  - Example: /oscar.nikhef.nl/mcprod/Role=production/Capability=NULL

In-depth view on VOMS (2)

- **The Attribute signing certificate MUST* be installed on each infrastructural machine that needs to verify VOMS Attributes**
  - This means that all entry-point middleware MUST* verify the VOMS Attributes
  - Installed in $VOMSDIR (default: “/etc/grid-security/vomsdir/”)
  - Though it seems similar to the CA RPMs distribution and installation, the amount of VOs on planet Earth will exceed the amount of CAs
    - VOMS certificates are normal {host|service|user} X.509 certificates
    - Usually expire each year
    - Will need timely renewal and redistribution

- **New mechanism in VOMS >1.7.0 makes the installation of the VOMS Issuer Certificates optional**
  - *More about this in more detail later in the slides*

***: these MUST are specific to EGEE/LCG as deployed today**

FQAN: Group structuring

- **Group structuring is expressed in the FQAN**
  - /<root group>/<subgroup>/…/<subgroup>
- **<root group> MUST be the name of the Virtual Organization**
- **Amount of subgroups is endless in an FQAN**
- **Group membership is compulsory and cannot be denied**
- **A member of a subgroup MUST be a member of the parent (sub)group**

  /oscar.nikhef.nl

  /oscar.nikhef.nl/g1

  /oscar.nikhef.nl/g1/sg1

  /oscar.nikhef.nl/g1/sg1/sg1.1

FQAN: Role

- **Roles are not organized in a hierarchical structure**
- **Roles are optional**
- **Ownership of a role is always associated to membership in a group**
- **If no specific role is held, the <role name> is NULL**
- **The member MUST be also a member of the group in which the Role is associated**

- **FQAN:**
  - <group name>/Role=<role name>/Capability=<capability name>

/oscar.nikhef.nl
/oscar.nikhef.nl/g1
/oscar.nikhef.nl/g1/Role=TheAdmin
/oscar.nikhef.nl/g1/sg1

FQAN: Capabilities

- **All rules that apply for a /Role apply also on /Capability**
- **The /Capability=<capability name> part is deprecated and will disappear in the future**
  - conforming applications SHOULD be able to handle FQANs where it is absent and SHOULD NOT rely on its presence.

Extensions

- **VOMS version > 1.7.0 allow for extensions to the existing fields**
  - Issuer Certificate:
    - This extension is meant to include the AA's public key certificate and the whole certificate chain leading to it, up to and excluding the CA certificate that is expected to be on the evaluator's machine (typically, the root CA)
    - If this extension is present, the evaluator MAY choose to use this certificate to verify the AC
  - Tags: *(VOMS version > 1.7.10)*
    - 'name' = 'value' pair plus 'qualifier' are linked to a policyAuthority
      - *The policyAuthority specifies which authority is the source of the enclosed tags*
    - Intended to provide a way to specify (arbitrary) attributes
    - 'name' AND 'value' AND 'qualifier' == unique

?

Sequences

Sequence of voms-proxy-init

- **voms-proxy-init --voms gin.ggf.org**
  - Having vomses file like:

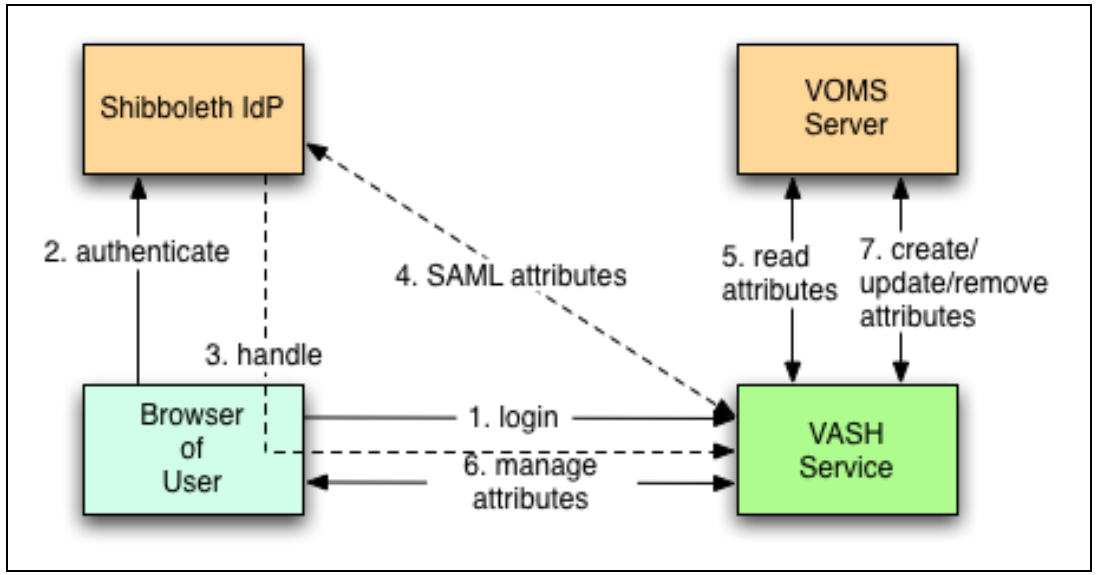
"gin.ggf.org" "kuiken.nikhef.nl" "15050" "/O=dutchgrid…/CN=kuiken.nikhef.nl" "gin.ggf.org"

  - Optionally: the voms-proxy-init command can be extended to request Roles to be added
- **Create temp. proxy for GSI connection to 'vomsd' on kuiken.nikhef.nl**
- **Perform GSI connection to 'vomsd'**
  - Performs the regular checks
  - vomsd uses the User DN (and Issuer DN) and searches the database for groups (and Roles (and Capabilities))
  - Constructing the VOMS ACs and signing the ACs
  - Sending back the signed attributes to the client
- **Create a new proxy certificate and include the returned VOMS ACs into the new proxy**

Sequence of voms-api (at grid service)

- **Grab the certificates (from the TLS connection or file)**
- **Walk through the certificates chain in search of the VOMS ACs**
  - On the first positive result, use that set of VOMS ACs
- **Try to parse the result and try to verify them**
  - Unless explicitly specified not to verify the signatures
  - Verify the signature against the signers public key
    - Search in the $VOMSDIR for installed certificate(s)
    - Perform regular checks on the found certificate(s)
      - *Path validation*
      - *Revocation lists check*
      - *Etc.*
    - Try to fit a working public key on the signatures
  - Create one structure of all the VOMS ACs found
- **Return the structure**

Coping with the 'Grid resource'



- **Problem:**
  - Shibboleth federation concept not applicable for grid! (each grid component would need to become a federation member)
- **2 Complementary Approaches**
  - SLCS & mediator service VASH
  - (Switch's phase 3…)

?

A vulnerability & solution

"The trust in one VOMS server can be exploited by the creation of the same FQANs issued by another VOMS Server"

Recipe & Solution

- **Recipe:**
  - Two VOMS servers
    - VOMS Server ‘A’ serves ‘/atlas’
    - VOMS Server ‘B’ servers ‘/cms’
  - One Grid resource (Computing Element or Storage Element)
  - Trust both VOMS Servers by their Public Keys on the resource
  - Then… without telling the ‘cms’ VOMS server could start issuing ‘/atlas/*’ FQANs
    - This would break the relationship a bit…
- **Solution**
  - Use VOMS core daemon version >= 1.6.8
  - Create subdirs in the $VOMSDIR named like the VO (the root-group)
    - Pub Key file: /etc/grid-security/vomsdir/atlas/kuiken.nikhef.nl.pem
    - Limiting the bounds of that key file by the namespace

?

The distribution problem

VOMS Cert Dist problems

The problem:

- **The X.509 Host certificate MUST* be installed on disk for each infrastructural machine that needs to verify VOMS Attributes**
  - All entry-point middleware MUST* verify the VOMS Attributes
  - That are a lot of machines…
  - Though it seems similar to the CA RPMs distribution and installation, the amount of VOs on planet Earth will exceed the amount of CAs
    - VOMS certificates are normal host certificates that usually expire each year and need redistribution to keep the server alive and kicking

Facts of life:

- **A lot of SysAdmins didn't know that the base64 encoded part (the gibberish at the end of the file) is the only thing that OpenSSL is concerned about and not the humanly readable decoded part**
  - cat kuiken.nikhef.nl.pem
    - Would show the usual output, stating that this is the kuiken server
  - openssl x509 –in kuiken.nikhef.nl.pem –text
    - can have a significantly different result!

***: these 'MUST's are specific to EGEE/LCG**

VOMS Cert Dist proposed solution (0)

Solution (the one-liner):

- **embedding the VOMS Host Certificate in the proxy VOMS AC in each user proxy**

VOMS Cert Dist proposed solution (1)

VOMS version < 1.7.0
Sequence of voms-proxy-init

- **voms-proxy-init --voms gin.ggf.org**
- **Create temp. proxy for GSI connection to 'vomsd' on kuiken.nikhef.nl**
- **Perform GSI connection to 'vomsd'**
  - Performs the regular checks
  - vomsd uses the User DN (and Issuer DN) and searches the database for groups (and Roles (and Capabilities))
  - Constructing the VOMS ACs and signing the ACs
  - Sending back the signed attributes to the client
- **Create a new proxy certificate and include the returned VOMS ACs into the new proxy**

VOMS version > 1.7.0
Sequence of voms-proxy-init (diff):

[…]

- **Perform GSI connection to 'vomsd'**
  […]
  - Constructing the VOMS ACs and signing the ACs
    - ***This now includes the signing certificate to be embedded into the VOMS Extensions field***
  - Sending back the signed attributes to the client

[…]

VOMS Cert Dist proposed solution (2)

VOMS version < 1.7.0

Sequence of voms-api (at grid-services)

- **Walk through the certificates chain in search of the VOMS ACs**
- **Try to parse the result and try to verify them**
  - Unless explicitly specified not to verify the signatures
  - Verify the signature against the signers public key
    - Search in the $VOMSDIR for installed certificates
    - Perform regular checks on the found certificate(s)
      - *Path validation*
      - *Revocation lists check*
      - *Etc.*
    - Try to fit a working public key on the signatures
  - Create one structure of all the VOMS ACs found
- **Return the structure**

VOMS version > 1.7.0

Sequence of voms-api (at grid-services)

[…]

- **Try to parse the result and try to verify them**

  […]
  - Verify the signature against the signers public key
    - Search VOMS ACs itself for their Issuer Certificate
    - Search for AC Issuer Cert policy file on localhost
    - If both are found:
      - *match the policy file against the issuer certificate*
    - if they not match
      - *FAIL verification*
    - else
      - *Search in the $VOMSDIR for installed certificates*
    - Try to fit a working public key on the signatures
    - Search VOMS ACs itself for their Issuer identity (not certificate)
    - Perform regular checks on the found certificate(s)
      - *Path validation*
      - *Revocation lists check*
  - Create one structure of all the VOMS ACs found
- **Return the structure**

?

The VO Naming Convention

Index.voms

- Detailed VO Name Information
- New Global VO Naming convention
- The solution
- What we did for GGF AuthZ workgroup
- The accepted VO Naming statement
- The document highlights

VO Name Information (1)

- **Allowed VO (and group/role name) characters:**
  - [a-zA-Z0-9-_\.]
  - In English:
    - VO names can start with a number
    - VO Names are alphanumeric and can also contain the characters minus/dash/hyphen, underscore and dot

- **The FQAN format is defacto standardized to the following format:**
  - Group(s) part:
    - /<VO Name> [[/<group 1>]/<subgroup N>]
      - *Where <VO Name> equals the root group which equals the VO Name*
  - Role part:
    - [/Role=<your role>]
  - Capability part (deprecated but still available):
    - [/Capability=<your capability>]
  - An FQAN is a concatenation of the Group(s), Role and Capability part

VO Name Information (2)

voms-proxy-info –all

subject : /O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo/CN=proxy

issuer : /O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo

identity : /O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo

type : proxy

strength : 512 bits

path : /tmp/x509up_u7381

timeleft : 11:59:19

VO : TEST

subject : /O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo

issuer : /O=dutchgrid/O=hosts/OU=nikhef.nl/CN=kuiken.nikhef.nl

attribute :
/TEST/0123456789012345678/Role=0123456789/Capability=NULL

attribute : /TEST/blaat/Role=NULL/Capability=NULL

attribute : /TEST/workshop/Role=NULL/Capability=NULL

attribute : /TEST/workshop_with_a_long_or_more_or_less_huge_name/Role=NULL/Capability=NULL

attribute : /TEST/blaat/test/Role=NULL/Capability=NULL

attribute :
/TEST/0123456789012345678/Role=NULL/Capability=NULL

timeleft : 11:59:18

VO Name Information (3)

- **In theory there is no limit to the names**
  - This MUST be honored in all middleware that uses FQANs
  - In reality the VOMS Database itself has a (practical) limitation to the length originating from the VOMS DB schema

- **The Group(s), Role and Capability parts currently have a database limited length of 255 characters each**
  - Which means 255 -1 characters are possible for a VO name at maximum because all group FQANs are prefixed with a slash
    - No (sub) groups can then be created within such string
  - The Role string (without “/Role=“) can be 255 characters
  - The Capability string (without the “/Capability=“) can be 255 characters

VO Name Information (4)

…which means that an FQAN can be:

Groups part = 255 characters
Role part = "/Role=" (6) + 255 = 261 chars
Capability part = "/Capability=" (12) + 255 = 267 chars

… as large as: 255 + 261 + 267 = 783 characters

New Global VO naming proposal

- **The Problem:**
  - No name (-space) control
  - Name clashes are starting to appear
    - FUSION and FUSION'
      - *first real name clash*
    - ATLAS as a name is used in many places
      - *One VO with different names*
    - Biomed vs. Bio Italy
      - *Two VOs same area of work, but unrelated*
- **The Solution:**
  - A hierarchical, extensible VO name space is needed

The DNS solution

- **Less confusion and less mix-ups**
- **The DNS scheme serves the same kind of purpose**

- **RFC 1034: Domain names - concepts and facilities**
  - The diff between how VOMS is deployed today and the RFC1034 are the following:
    - RFC 1034 does not allow underscores, VOMS does
  - You SHOULD use 7-bit ASCII characters
    - a-zA-Z[a-zA-Z0-9-\.]*

Time for GIN?

The VO *Grid Interoperability Now* is the first to be created in the new scheme

gin.ggf.org

The VO Naming statement

The **VO name is a string**, used to represent the VO in all interactions with grid software, such as in expressions of policy and access rights.

The **VO name MUST be formatted as a subdomain name as specified in RFC 1034** section 3.5. **The VO Manager** of a VO using a thus-formatted name **MUST be entitled to the use of this name**, when interpreted as a name in the Internet Domain Name System.

This entitlement MUST stem either from a direct delegation of the corresponding name in the Domain Name System by an accredited registrar for the next-higher level subdomain, or from a direct delegation of the equivalent name in the Domain Name System by ICANN, **or from the consent of the administrative or operational contact of the next-higher equivalent subdomain name** for that VO name that itself is registered with such **an accredited registrar**.

Considering that RFC1034 section 3.5 states that both upper case and lower case letters are allowed, but no significance is to be attached to the case, but that today the software handling VO names may still be case sensitive, **all VO names MUST be entirely in lower case**.

?

AuthZ & Mapping

The tools

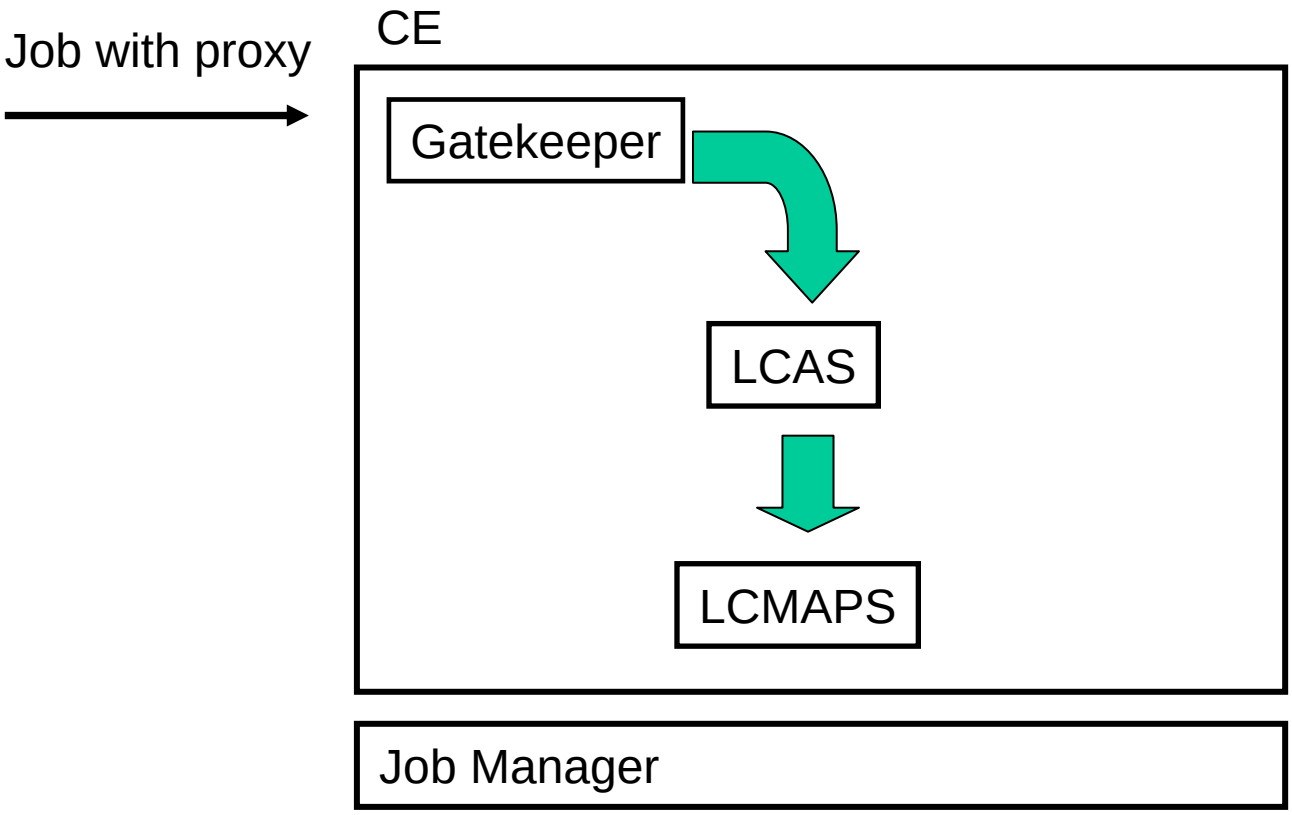
Site Security with VOMS aware tools

- **Current site security mechanisms on LCG-2**
  - JAVA: java-security
  - C: grid-mapfile & LCAS/LCMAPS
    - I only know the LCAS/LCMAPS very well so I'll continue with this…

- **LCAS/LCMAPS can be used for AuthZ and user mapping functionality in the edg-gatekeeper and edg-gridFTP**
  - Currently available as LCG software
  - edg-gridFTP with LCAS/LCMAPS call-out has been tested by LCG
  - Near future GT-4 interface to LCAS and LCMAPS will be available

- **PRIMA, SAZ and GUMS**
  - Prima is the library that dispatches the credential checks to SAZ and the identity mapping to GUMS
  - GUMS uses an extended SAML protocol

Example CE workflow



LCAS

- **Local Centre Authorization Service**
  - Pluggable authorization framework
  - grid-mapfile
  - Plug-in for VOMS
    - Uses VOMS API
    - AuthZ policy in GACL format (or grid-mapfile)
    - Convenience tool to convert grid-mapfile into GACL format:

      `edg-lcas-voms2gacl`
  - Ban list
  - Other conditions that can be true or false (like fabric opening times)
  - Execution white list
  - Extendable
    - Shibboleth authorization is in the making…

LCMAPS

- **Local Credential MAPping Service**
  - Pluggable identity mapping framework
  - Provides local credentials needed for jobs in fabric
    - Mapping based on user identity, VO affiliation, site-local policy
    - Supports standard UNIX credentials (uid & gid)
    - Poolaccounts & Poolgroups
  - Fine grained mapping driven by the local site policy and VOMS-proxy extensions
  - Can also be use AFS Tokens (fixed by INFN)
  - LDAP user directory update
  - The Job Repository is a plug-in that can store the user, VO and job information, with regards to its mapping, into a database
    - Can be used as accounting information base
  - Extendable
    - The framework is being used by G-PBox (INFN-CNAF) to manipulate the result of an identity mapping by the inclusion of a higher level policy that will alter the decision made by a cluster scheduler
    - The Shib. developers from Switch will also do user authz & mapping based upon the Shibboleth identity embedded into the VOMS ACs by VASH.

AuthZ & Mapping

LCMAPS: How we perform a mapping

VOMS mapping with LCMAPS

- **Support for multiple VOs per user (and thus multiple UNIX groups)**
- **Boundary conditions**
  - Has to run in privileged mode
  - Has to run in process space of incoming connection (for *fork* jobs)
- **Extendable…**

- **The most essential VOMS plug-ins can do:**
  - Parsing of the proxy certificate for VOMS attributes
  - Determining how to map an FQAN to a UNIX group
    - Described in the vo-mapfile (or extended grid-mapfile) and groupmapfile
    - Plug-ins can look for localgroups & poolgroups
  - Determining which (VOMS) poolaccount it should lease for the (set of) FQANs
    - Described in vomapfile or extended grid-mapfile

LCMAPS VOMS Parsing in more detail

- **We gather the FQANs in the framework and make it available for all the plugins**
  - Execute the 'voms poolgroup' plugin (optional)
    - Search the groupmapfile for matches like:
      "<FQAN>" .grp"

Personal detail:

I didn't like to add this plugin, because we're mapping -multiple- users to -one- Unix account.

    - Store the mapp[illegible]ffective) Unix User IDs in framework
  - Execute the 'voms localaccount' plugin
    - Search the grid-mapfile or vomapfile for matches like:
      "<FQAN>" thataccount"
    - Store the mapping, (effective) Unix User IDs in framework

LCMAPS VOMS Parsing in more detail

- **The first FQAN in the VOMS Attributes will be mapped to the Primary Unix Group ID**
- **Multiple Primary Unix Group IDs are not allowed**
  - It can happen that two VOMS plugins try to add their first found mapping to the LCMAPS framework. That's discarded and solved when they try to push the information.
- **All the GIDs will be added as Secondary Unix Group IDs**
  - Gives you the rights to perform file access with all FQAN equivalent Unix Groups of synchronous with tools like gridFTP
  - The enforced GIDs will be in the order of their numerical representation
  - No GID is added twice
    - logical privilege squash without information loss
    - And … in the beginning we had RH 7.3.x with NGROUPS set to 32

AuthZ & Mapping

Examples

Example extended grid-mapfile

- **This extended grid-mapfile contains information for the VOMS poolaccounting**
  - This mapping will be determined by the **first** VOMS attribute that you have in the VOMS proxy

```
[…]
"/O=dutchgrid/O=users/O=nikhef/CN=Jeffrey Templon" templon
"/O=dutchgrid/O=users/O=nikhef/CN=Martijn Steenbakkers" .test
"/O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo" okoeroo
"/O=edgtutorial/O=users/O=edg-tutorial/CN=Grid pupil 20" davidg
"/fred/*" .husband
"/wilma/pebbles" martijn
"/wilma/*" .wife
```

Example groupmapfile

- **This groupmapfile handles four VOs**
  - Note: these lines are *not* FQANs but look and feel like them

```
[okoeroo@asen okoeroo]$ cat /etc/grid-security/groupmapfile
# Example groupmapfile:

# Users with the exact VO-group info "/fred/Role=husband"
# will be added to the local group "fredje"
"/fred/*" fredje

# All users from VO wilma will be added to the allocated pool group "pool[1-9]*"
"/wilma/*" .pool
```

Example job-run

- **Job with VOMS (Exec: /usr/bin/id –a)**
  - With FQANs scenario A:
    - /fred
  - With FQANs attributes scenario B:
    - /wilma
  - With FQANs attributes scenario C:
    - /wilma/Role=wife
    - /wilma
  - With FQANs attributes scenario D:
    - /wilma/Role=wife
    - /wilma
    - /fred
  - With FQANs attributes scenario E:
    - /fred
    - /wilma/Role=wife
    - /wilma

- **VOMS Job Results:**
  - A: uid=4101(husband001) gid=2100(fredje)
  - B: uid=4201(wife001) gid=4001(pool001)
  - C: uid=4202(wife002) gid=4002(pool002) groups=4001(pool001)
  - D: uid=4203(wife003) gid=4002(pool002) groups=4001(pool001), 2100(fredje)
  - E: uid=4102(husband002) gid=2100(fredje) groups=4002(pool002), 4001(pool001)

?

GUMS vs. LCMAPS

General Approach

GUMS

- Separation of the decision point from the client.
- GUMS is a centralized Policy Decision Point that can serve several clients.
- On the client side, it has the PRIMA libraries. Plugins for GT2, GT4 and gLExec exist.

LCMAPS

- All-in-one solution
- LCMAPS is both the decision point and client library
- The policies are stored in a file system accessible by the LCMAPS framework.

Approach to Authentication

GUMS

- The client verifies the credentials and extracts the DN and the primary FQAN
- The DN+FQAN are sent to GUMS over server-side-only HTTPS using the SAML protocol
- GUMS returns a SAML response + XACML obligations (non standard), like UserID=XYZ
- The current implementation of PRIMA does not validate the validity of FQANs.
  - The GUMS database pre-stores the DN to FQANs group affiliations thus a user can't go out of this boundary

LCMAPS

- Verifies the user credentials, both the DN and FQANs.
- The LCMAPS framework holds all credentials
  - either from the input (GK, glexec, gridFTP)
  - Or by using one of the acquisition plugins that scavenge credentials
- The LCMAPS framework extracts and verifies the VOMS information on extraction from the ACs
  - The LCMAPS framework doesn't do any checks itself, for this you'll have the lcmaps_verify_proxy plugin to do the job
- When LCMAPS is executed the successful mapping is performed on the current process itself
- Credentials passed to LCMAPS can be accumulated and verified; a mapping flows out of that when sufficient credentials are present and verified
- The mapping granularity is in control of the sysadmin.
  - No need to sync with a VOMS server
  - All authZ and mapping can be done according to a local configuration
  - No need to construct a relational database (reconstruction of the VOMS DB on each site) of all the users of all VOs that wish to have a potential mapping on a site

Implementation

GUMS

- **GUMS is a web service. Java code inside Tomcat.**
- **Typically, GUMS needs:**
  - Tomcat instance
  - MySQL
- **PRIMA available in both C (for GT2 and gLExec) and Java (GT4) implementations.**
  - No requirements for PRIMA (just the code).

LCMAPS

- **Everything is C based.**
- **Policy store today:**
  - filesystem
- **Cross-node mapping consistency can be implemented via NFS lock mechanism**
- **Traceability via JobRepository DB (an optional plug-in)**

Plug-in architecture

GUMS

- **Plug-in based with Java-based plug-ins.**
- **Possibility to add new Classes to add functionality**
- **All configuration held in a single, XML file. The plug-ins configured here too, as attributes of plug-in tags.**

LCMAPS

- **Plug-in based.**
  - Plug-ins are shared libraries.
  - One global text file to list the shared libraries to include.
- **Each plugin is initialized from the lcmaps.db config file.**
  - If needed (like the database password for the Job Repository) plugins could need their own config files.

Policy handling

GUMS

- **By class interfaces: Five main types of class interfaces:**
  - storage
    - database (JDBC) - most used, support both static and dynamic mappings
  - User groups
    - manual - forced one to one mapping
    - VOMS group accounts- load every 6 hours all DN/FQANs from a VOMS and maps them all into one UID
    - VOMS pool account - load every 6 hours all DN/FQANs from a VOMS and maps them all into pool accounts (all different)
    - LDAP
  - host to group mapping
    - given a host expression (like "fcdf*.fnal.gov") list of groups to map to.
  - group to account mapping
    - given a group, one or more account mappers that will return the local account to map to (the first one to return a hit).
  - user group
    - Used by group to account mapping to verify user group/VO membership given user DN+FQAN.
  - account mapper
    - Used by group to account mapping to return account name given user DN.

LCMAPS

- **The policy handling in LCMAPS is based around the plug-ins that it will need to execute**
  - Which means the plug-ins can control the course for the mapping
- **Quite simple state machine:**

<policy name>:

plugin1 (execute this plugin) -> plugin2 (if plugin1 is successfull) | plugin3 (if plugin1 failed)

plugin2 (execute plugin2) -> plugin3 (execute plugin3 when plugin2 is successfull)

Example policies

GUMS

```
<gums>
        <persistenceFactories>
          <persistenceFactory
            name="mysql"
            className="gov.bnl.gums.hibernate.HibernatePersistenceFactory"
            hibernate.connection.driver_class="com.mysql.jdbc.Driver"
            hibernate.dialect="net.sf.hibernate.dialect.MySQLDialect"
            hibernate.c3p0.min_size="3"
            hibernate.c3p0.max_size="20"
            hibernate.c3p0.timeout="180"
            hibernate.connection.url="jdbc:mysql://localhost:49251/GUMS_1_1"
            hibernate.connection.username="****"
            hibernate.connection.password="****"
            hibernate.connection.autoReconnect="true"/>
          </persistenceFactories>
          <groupMappings>
            <groupMapping name="atlas">
              <userGroup
                className="gov.bnl.gums.LDAPGroup"
                server="grid-vo.nikhef.nl"
                query="ou=lcg1,o=atlas,dc=eu-datagrid,dc=org"
                persistenceFactory="mysql"
                name="atlas"/>
              <accountMapping
                className="gov.bnl.gums.GroupAccountMapper"
                groupName="usatlas1"/>
            </groupMapping>
            <groupMapping name="vomsAtlas">
              <userGroup
                className="gov.bnl.gums.VOMSGroup"
                url="https://lcg-voms.cern.ch:8443/voms/atlas/services/VOMSAdmin"
                persistenceFactory="mysql"
                sslCertfile="/etc/grid-security/gumscert.pem"
                sslKey="/etc/grid-security/gumskey.pem"
                matchFQAN="ignore"
                acceptProxyWithoutFQAN="true"
                voGroup="/atlas"
                name="vomsatlas"/>
              <accountMapping
                className="gov.bnl.gums.GroupAccountMapper"
                groupName="usatlas1"/>
            </groupMapping>
             <groupMapping
               name="cdfPool"
               accountingVo="cdf"
               accountingDesc="CDF">
               <userGroup
                 className="gov.bnl.gums.VOMSGroup"
                 url="https://voms.cnaf.infn.it:8443/voms/cdf/services/VOMSAdmin"
                 persistenceFactory="mysql"
                 name="osgcdf"
                 voGroup="/cdf"
                 sslCertfile="/etc/grid-security/gumscert.pem"
                 sslKey="/etc/grid-security/gumskey.pem"
                 matchFQAN="ignore"
                 acceptProxyWithoutFQAN="true"/>
               <compositeAccountMapping>
                 <accountMapping
                   className="gov.bnl.gums.AccountPoolMapper"
                   persistenceFactory="bnl"
                   name="bnlPool.cdf"/>
               </compositeAccountMapping>
             </groupMapping>
          </groupMappings>
          <hostGroups>
            <hostGroup
              className="gov.bnl.gums.CertificateHostGroup"
              cn="cdfonly*.fnal.gov"
              groups="cdfPool"/>
           <hostGroup
              className="gov.bnl.gums.CertificateHostGroup"
              cn="osg*.fnal.gov"
              groups="cdfPool,vomsAtlas"/>
          <hostGroup
              className="gov.bnl.gums.CertificateHostGroup"
              cn="lcg*.fnal.gov"
              groups="cdfPool,atlas"/>
          </hostGroups>
      </gums>
```

LCMAPS

voms:

- vomslocalgroup -> vomspoolgroup
- vomspoolgroup -> vomspoolaccount | vomspoolaccount
- vomspoolaccount -> posix_enf

legacy:

- localaccount -> posix_enf | poolaccount
- poolaccount -> posix_enf

Mapping service

GUMS

- **Maps DN+primary FQAN into a UID and optionally a GID, too.**
- **PRIMA passes these values to the calling client.**

LCMAPS

- **Maps DN+primary FQAN into (UID,GID)**
  - All secondary FQANs are mapped to secondary GIDs
- **LCMAPS is the calling client itself**

Proposals for future improvements

GUMS

- **Convert GUMS to use XACML. This way we can relinquish PRIMA and use standard XACML libraries.**
- **If possible, integrate GUMS functionality into the Globus CAS.**

LCMAPS

- **Create central site AuthZ and Mapping service**
- **Split the VOMS Acquisition from the LCMAPS framework (like it was 2 years ago)**
- **Try to find a more common way to store credentials in the framework**
  - treat them as arbitrary sources for mappings.
  - In this way we can support:
    - Globus CAS
    - Shibboleth
    - VOMS
    - other OIDs and any other possible type of credential.

Bottomline…

- **GUMS+PRIMA performs the same task as LCMAPS but has a quite different design**
  - Due to different views and impacts of that design we can't use GUMS+PRIMA directly (at least not) on the European sites
    - Current GUMS uses mkgridmap-style VO member propagation based on DN string matching only (not signed assertions)
    - Not going to convert LCMAPS to work with GUMS natively within a foreseeable future
    - LCMAPS (and LCAS) will also sport a central AA/mapping service
    - Wire protocol compatibility is more viable route
  - GUMS may alter its design to be more compatible
    - Needs a internal reimplementation on the mapping sequences
  - Plug-ins created by a 3rd-parties (like GPBox and AFS plug-in, and the upcoming Shib plug-in)
    based on the LCMAPS interfaces and will need to be re-implemented to be used in a GUMS environment

?

Future

- **VOMS & LCAS/LCMAPS will continue as part of gLite**
  - VOMS Parser in JAVA (before only in C/C++)
    - also a creator of proxies is now available in Java
  - Creating a site central mapping service as part of the LCAS/LCMAPS framework
    - Possibly a gsoap implementation
  - Support towards new log file formatting rules
  - Use a standard configuration format like XACML

Acknowledgments

- **VOMS and VOMS Admin is supported by INFN-CNAF**
  - Team of Vincenzo Ciaschini, Valerio Venturi and Andrea Ceccanti
- **glexec, LCAS and LCMAPS (and most of the basic plugins) are supported by NIKHEF**
  - Team of David Groep, Oscar Koeroo and Gerben Venekamp

?